CMPE150.05

08.12.2018

CMPE150 ASSIGNMENT 2

Problem description: In this assignment we were asked to write a Java program for a 4x4 tic-tac-toe game played against the computer, with a load file functionality included.

Problem solution: In order to solve this problem, I used 6 for-loops in total and 3 static methods (excluding main), 2 of them returning ints and 1 of them returning a boolean. These were, in relative order of increasing complexity, as follows:

1) **hasEnded:** This method takes as argument the String board (which is the four lines of the board written side by side on one line) and returns a boolean. It checks whether the game has ended or not, by testing every possible winning combination and seeing if there are 3 of the same letter ('X' or 'O') at those positions. If yes, returns true (declaring that the game has ended.) Otherwise, returns false.
2) **whoWon:** This method is very similar to hasEnded, its job is to check who won the game after the game has been declared as ended. Again, it checks all the winning combinations to see if there are 3 of the same character at those positions. If yes, it checks whether that character is equal to the character chosen by the player at the start of the game (meaning, whether it's the player who won or the computer.) If the player has won, it returns 0. Otherwise, it returns 1. The tie case is handled by the playTheGame method, which will be explained next.
3) **main:** Starts out by declaring a Scanner that will read from the console and then initializing some variables to essentially "0" values, (ints to 0, Strings to empty strings). These variables will later be updated and some will eventually be used as arguments to call the playTheGame method. It then greets the user and asks whether they want to create a new game or load a savefile. It awaits user input and reprompts the user to enter valid values as long as they don't enter either L or C.
   - If the user chooses C, it asks for the symbol the player wants to use and reprompts the user as long as the input is not X or O.
   - If the user chooses L, it asks for the name of the file to be loaded. It reprompts the user as long as the file does not exist on the disk. Then it performs four tests on the file using four different scanners. During these four tests, it checks whether the file has 4 lines that are each 17 characters long, and whether all the lines contain only the characters: {|, , E, X, O}. If any of these tests fail, the user is asked to re-enter a valid filename. Meanwhile, it counts the total number of X's and O's in the file in order to determine who will go first. (O if there are more X's vice versa). If the file passes all the tests, it asks the user to choose their symbol.

   Then there's a do-while loop (so that the game executes at least once, then keeps going as long as the user enters "Y" when asked if they want to play again. Inside the do-while loop, the method playTheGame is called with 5 arguments. There's also a switch-case block for the three possible outcomes of a game: 1 for computer win, 0 for player win and -1 for a tie. At the end of a game, if the user enters "N", the numbers of times the player and the computer won are printed, the program says goodbye and terminates.

4) **playTheGame:** This is essentially the bulk of the program. It takes 5 arguments as input: String choice (indicating whether the player has opted to create a new game or load a savefile), Scanner console (generic scanner to read user inputs from the console), String playerChoice (indicating whether the player has chosen to play as 'X' or 'O'), String filename (the name of the file to be loaded if the player has chosen to load a file) and String starter (with values "X" or "O", this shows which symbol will get to go first, determined in the main method). It returns an integer (1 if the computer won, 0 if the player won and -1 if it's a tie.). It starts out with a giant if-else construct depending on whether the user chose C or L.

- If they chose C, it assigns the respective symbols to both the player and the computer. It uses a random number generator that generates either 0 or 1 to determine who makes the first move.
  - If the player gets to start, it prompts the user for coordinates (integer values between 1 and 4, both inclusive.) It does not accept any other types of input and handles that case with a try-catch block. After it has taken valid coordinates, it checks whether the cell with the specified coordinates is empty. If yes, it modifies that cell by inserting the player's symbol instead of an E. Then, it generates a pair of random coordinates for the computer and does the insertion similarly. When this cycle of 2 steps is over, the String board (which is a String that contains the four lines of the board all side by side) is updated with the newly-changed characters, and the new state of the gameboard is printed out on the screen. At the end of each iteration, the program checks whether any E's remain on the board, because if no one has won yet there's no empty cell left on the board, it means that the game has ended as a tie. Therefore, in that case the method terminates by returning -1. This whole cycle is inside a while loop which keeps executing as long as the hasEnded method returns false. Once it returns true, the program immediately terminates and returns the result of the whoWon method.
  - If the computer starts, the process is still roughly the same, except the computer plays much more competitively. Since in 4x4 tic-tac-toe, the one who starts has a 100% chance of winning if they play the right moves, this portion of the program is written in such a way that the computer makes those right moves. It always starts by playing the move (2,2) and then it makes moves accordingly with the player's moves in order to counter them. At the end of the third round, the computer always wins.
- If they chose L, the process is again roughly the same with only two major exceptions:
  - 1. We need to perform one more test on the file in addition to the ones performed in the main method. Even though the board may have been "valid" so to speak, it might contain a board that has already been filled out completely but in which no side has won (a tie game), or one that contains a game that has already ended. In that case, the hasEnded & whoWon methods correctly handle the case where the file contains a board where the game has ended. But we need another test to handle the case where neither side has won even though no empty spaces remain on the board. For that, we check if there are any empty cells in the board. If that's not the case, since the hasEnded method is still returning false but no viable moves remain, the game HAS actually ended but in a tie.
  - 2. If the computer starts, this time the computer's moves are handled with a random number generator (the computer is unable to play competitively). When the game starts out with an empty board, it's easy to dictate the winning moves the computer

has to make. But when we load a file, determining a winning strategy is not as easy. So, when a file is loaded and the computer goes first, it just plays randomly.

Implementation:

```
package ECK2018400018;

import java.util.Scanner;        import java.io.*;
import java.util.InputMismatchException;
import java.util.Random;

public class ECK2018400018
{
        // checks whether the game has ended or not, by testing every possible winning combination and
seeing if there are 3 of the same letter at those positions (except for the letter 'E')
        public static boolean hasEnded(String board)
        {
                if (    (board.charAt(2) == board.charAt(6) && board.charAt(6) == board.charAt(10) &&
board.charAt(2) != 'E') ||
                          (board.charAt(6) == board.charAt(10) && board.charAt(10) == board.charAt(14) &&
board.charAt(6) != 'E') ||
                        (board.charAt(19) == board.charAt(23) && board.charAt(23) == board.charAt(27) &&
board.charAt(19) != 'E') ||
                        (board.charAt(23) == board.charAt(27) && board.charAt(27) == board.charAt(31) &&
board.charAt(23) != 'E') ||
                        (board.charAt(36) == board.charAt(40) && board.charAt(40) == board.charAt(44) &&
board.charAt(36) != 'E') ||
                        (board.charAt(40) == board.charAt(44) && board.charAt(44) == board.charAt(48) &&
board.charAt(40) != 'E') ||
                        (board.charAt(53) == board.charAt(57) && board.charAt(57) == board.charAt(61) &&
board.charAt(53) != 'E') ||
                        (board.charAt(57) == board.charAt(61) && board.charAt(61) == board.charAt(65) &&
board.charAt(57) != 'E') ||
                          (board.charAt(2) == board.charAt(19) && board.charAt(19) == board.charAt(36) &&
board.charAt(2) != 'E') ||
                        (board.charAt(19) == board.charAt(36) && board.charAt(36) == board.charAt(53) &&
board.charAt(19) != 'E') ||
                          (board.charAt(6) == board.charAt(23) && board.charAt(23) == board.charAt(40) &&
board.charAt(6) != 'E') ||
                        (board.charAt(23) == board.charAt(40) && board.charAt(40) == board.charAt(57) &&
board.charAt(23) != 'E') ||
                        (board.charAt(10) == board.charAt(27) && board.charAt(27) == board.charAt(44) &&
board.charAt(10) != 'E') ||
                        (board.charAt(27) == board.charAt(44) && board.charAt(44) == board.charAt(61) &&
board.charAt(27) != 'E') ||
                        (board.charAt(14) == board.charAt(31) && board.charAt(31) == board.charAt(48) &&
board.charAt(14) != 'E') ||
                        (board.charAt(31) == board.charAt(48) && board.charAt(48) == board.charAt(65) &&
board.charAt(31) != 'E') ||
                          (board.charAt(2) == board.charAt(23) && board.charAt(23) == board.charAt(44) &&
board.charAt(2) != 'E') ||
                          (board.charAt(6) == board.charAt(27) && board.charAt(27) == board.charAt(48) &&
board.charAt(6) != 'E') ||
                        (board.charAt(10) == board.charAt(23) && board.charAt(23) == board.charAt(36) &&
board.charAt(10) != 'E') ||
                        (board.charAt(14) == board.charAt(27) && board.charAt(27) == board.charAt(40) &&
board.charAt(14) != 'E') ||
                        (board.charAt(19) == board.charAt(40) && board.charAt(40) == board.charAt(61) &&
board.charAt(19) != 'E') ||
                        (board.charAt(24) == board.charAt(44) && board.charAt(44) == board.charAt(65) &&
board.charAt(24) != 'E') ||
                        (board.charAt(27) == board.charAt(40) && board.charAt(40) == board.charAt(53) &&
board.charAt(27) != 'E') ||
                        (board.charAt(31) == board.charAt(44) && board.charAt(44) == board.charAt(57) &&
board.charAt(31) != 'E'))
                {
                        return true;
                }
```

```
                return false;
        }

        // when the method hasEnded returns true, this method checks who won. returns 0 if the player has
won, returns 1 if the computer has won. the tie case is handled inside the playTheGame method
        public static int whoWon(String board, char playerChoiceChar)
        {
                if (    (board.charAt(2) == board.charAt(6) && board.charAt(6) == board.charAt(10) &&
board.charAt(2) == playerChoiceChar) ||
                                (board.charAt(6) == board.charAt(10) && board.charAt(10) ==
board.charAt(14) && board.charAt(6) == playerChoiceChar) ||
                                        (board.charAt(19) == board.charAt(23) && board.charAt(23) ==
board.charAt(27) && board.charAt(19) == playerChoiceChar) ||
                                        (board.charAt(23) == board.charAt(27) && board.charAt(27) ==
board.charAt(31) && board.charAt(23) == playerChoiceChar) ||
                                        (board.charAt(36) == board.charAt(40) && board.charAt(40) ==
board.charAt(44) && board.charAt(36) == playerChoiceChar) ||
                                        (board.charAt(40) == board.charAt(44) && board.charAt(44) ==
board.charAt(48) && board.charAt(40) == playerChoiceChar) ||
                                        (board.charAt(53) == board.charAt(57) && board.charAt(57) ==
board.charAt(61) && board.charAt(53) == playerChoiceChar) ||
                                        (board.charAt(57) == board.charAt(61) && board.charAt(61) ==
board.charAt(65) && board.charAt(57) == playerChoiceChar) ||
                                          (board.charAt(2) == board.charAt(19) && board.charAt(19) ==
board.charAt(36) && board.charAt(2) == playerChoiceChar) ||
                                        (board.charAt(19) == board.charAt(36) && board.charAt(36) ==
board.charAt(53) && board.charAt(19) == playerChoiceChar) ||
                                          (board.charAt(6) == board.charAt(23) && board.charAt(23) ==
board.charAt(40) && board.charAt(6) == playerChoiceChar) ||
                                        (board.charAt(23) == board.charAt(40) && board.charAt(40) ==
board.charAt(57) && board.charAt(23) == playerChoiceChar) ||
                                        (board.charAt(10) == board.charAt(27) && board.charAt(27) ==
board.charAt(44) && board.charAt(10) == playerChoiceChar) ||
                                        (board.charAt(27) == board.charAt(44) && board.charAt(44) ==
board.charAt(61) && board.charAt(27) == playerChoiceChar) ||
                                        (board.charAt(14) == board.charAt(31) && board.charAt(31) ==
board.charAt(48) && board.charAt(14) == playerChoiceChar) ||
                                        (board.charAt(31) == board.charAt(48) && board.charAt(48) ==
board.charAt(65) && board.charAt(31) == playerChoiceChar) ||
                                          (board.charAt(2) == board.charAt(23) && board.charAt(23) ==
board.charAt(44) && board.charAt(2) == playerChoiceChar) ||
                                          (board.charAt(6) == board.charAt(27) && board.charAt(27) ==
board.charAt(48) && board.charAt(6) == playerChoiceChar) ||
                                        (board.charAt(10) == board.charAt(23) && board.charAt(23) ==
board.charAt(36) && board.charAt(10) == playerChoiceChar) ||
                                        (board.charAt(14) == board.charAt(27) && board.charAt(27) ==
board.charAt(40) && board.charAt(14) == playerChoiceChar) ||
                                        (board.charAt(19) == board.charAt(40) && board.charAt(40) ==
board.charAt(61) && board.charAt(19) == playerChoiceChar) ||
                                        (board.charAt(24) == board.charAt(44) && board.charAt(44) ==
board.charAt(65) && board.charAt(24) == playerChoiceChar) ||
                                        (board.charAt(27) == board.charAt(40) && board.charAt(40) ==
board.charAt(53) && board.charAt(27) == playerChoiceChar) ||
                                        (board.charAt(31) == board.charAt(44) && board.charAt(44) ==
board.charAt(57) && board.charAt(31) == playerChoiceChar))
                {
                        return 0;
                }

                return 1;
        }

        public static int playTheGame(String choice, Scanner console, String playerChoice, String
filename, String starter) throws FileNotFoundException, InputMismatchException
        {
                // player decided to create a new game
                if (choice.equals("C"))
                {
```

```
                        // assigns their respective letters to the player and the computer
                        String computerChoice =  playerChoice.equals("X") ? "O" : "X"; // computer's
letter
                        String playerChoiceInit = playerChoice; // keep a copy of the player's letter in
another variable
                        String computerChoiceInit = computerChoice; // keep a copy of the computer's
letter as well
                        char playerChoiceChar = playerChoice.charAt(0); // player's letter as a char

                        System.out.println("You are player " + playerChoice + " and the computer is
player " + computerChoice + "." );

                        Random random = new Random(); // a random number generator, will be used in
deciding who gets to go first and for the computer's moves

                        int whoGetsToGo = random.nextInt(2); // random int, if 0 the player starts, if 1
the computer starts

                        // the case when the player starts
                        if (whoGetsToGo == 0)
                        {
                                System.out.println("You will start.");

                                // how a default line looks, and the four lines of a board which are
initially set as equal to the default line
                                String defaultLine = "| E | E | E | E |";        String line1 =
defaultLine;     String line2 = defaultLine;      String line3 = defaultLine;      String line4 =
defaultLine;
                                String board = line1 + line2 + line3 + line4; // the whole board in one
line

                                for (int i = 0; i < 4; i++) // counts up to 4
                                {
                                        System.out.println(defaultLine);
                                }

                                // keep the game going as long as hasEnded returns false
                                while(!hasEnded(board))
                                {
                                        System.out.println("Enter coordinates:");

                                        int yCoordinate = 0;     int xCoordinate = 0; // the y coordinate
and the x coordinate that will be read from the user with a scanner, initialized to 0

                                        // warns user if they don't enter two integers (e.g. when they
try to enter a string instead)
                                        do
                                        {
                                                try
                                                {
                                                        yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
                                                }
                                                catch (InputMismatchException e)
                                                {
                                                        System.out.println("Wrong input! Try
again:");
                                                        console.nextLine();
                                                }
                                        } while (yCoordinate == 0 || xCoordinate == 0);

                                        // warns user if they enter a wrong coordinate (larger than 4 /
smaller than 1 / a non-empty cell)
                                        while ((xCoordinate > 4 || yCoordinate > 4) || board.charAt(-2 +
(4 * xCoordinate) + (17 * (yCoordinate - 1))) != 'E')
                                        {
                                                System.out.println("Wrong input! Try again:");
                                                yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
```

```
                                        }

                                        // a counter that counts up to 2 (to do the line-editing process
twice: one for the player and one for the computer.)
                                        for (int i = 0; i < 2; i++, playerChoice = computerChoiceInit)
                                        {
                                                // the line-editing process. using switches, this whole
block of code edits lines on the board using string methods. it's ugly but I made it this way since we
couldn't use arrays.
                                                switch (xCoordinate)
                                                {
                                                        // the outermost switch cases check the x
coordinate. for example, if it's 1, the 2nd character in each line will be modified. if it's 2, the 6th
character, and so on.
                                                        case 1:
                                                        switch (yCoordinate)
                                                        {
                                                                // the inner switch cases check the y
coordinate. for example, if it's 1, line1 will be edited; if it's 2, line 2 will be edited etc.
                                                                case 1:
                                                                line1 = line1.substring(0,2) +
playerChoice + line1.substring(3);
                                                                break;

                                                                case 2:
                                                                line2 = line2.substring(0,2) +
playerChoice + line2.substring(3);
                                                                break;

                                                                case 3:
                                                                line3 = line3.substring(0,2) +
playerChoice + line3.substring(3);
                                                                break;

                                                                case 4:
                                                                line4 = line4.substring(0,2) +
playerChoice + line4.substring(3);
                                                                break;
                                                        }
                                                        break;

                                                        case 2:
                                                        switch (yCoordinate)
                                                        {
                                                                case 1:
                                                                line1 = line1.substring(0,6) +
playerChoice + line1.substring(7);
                                                                break;

                                                                case 2:
                                                                line2 = line2.substring(0,6) +
playerChoice + line2.substring(7);
                                                                break;

                                                                case 3:
                                                                line3 = line3.substring(0,6) +
playerChoice + line3.substring(7);
                                                                break;

                                                                case 4:
                                                                line4 = line4.substring(0,6) +
playerChoice + line4.substring(7);
                                                                break;
                                                        }
                                                        break;

                                                        case 3:
                                                        switch (yCoordinate)
                                                        {
```

```
                                                        case 1:
                                                        line1 = line1.substring(0,10) +
playerChoice + line1.substring(11);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,10) +
playerChoice + line2.substring(11);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,10) +
playerChoice + line3.substring(11);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,10) +
playerChoice + line4.substring(11);
                                                        break;
                                                }
                                                break;

                                                case 4:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,14) +
playerChoice + line1.substring(15);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,14) +
playerChoice + line2.substring(15);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,14) +
playerChoice + line3.substring(15);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,14) +
playerChoice + line4.substring(15);
                                                        break;
                                                }
                                                break;
                                        }

                                        // change the x and y coordinates randomly (but to valid
values – i.e. empty cells) for the computer's move
                                        do
                                        {
                                                xCoordinate = random.nextInt(4) + 1; yCoordinate
= random.nextInt(4) + 1;
                                        }
                                        while (board.charAt(-2 + (4 * xCoordinate) + (17 *
(yCoordinate - 1))) != 'E');
                                }

                                board = line1 + line2 + line3 + line4; // update the single-line
board string with the new, modified versions of lines

                                // print out the board after the moves have been made
                                System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);

                                // if at any time, there are no empty cells left on the board,
return -1 to declare a tie
                                if (!board.contains("E"))
```

```
                                        {
                                                return -1;
                                        }

                                        playerChoice = playerChoiceInit;
                                }

                                // when hasEnded is true, and the while exits, this method (playTheGame)
will return the value of whoWon (that is, 0 if the player won and 1 if the computer won)
                                return (whoWon(board, playerChoiceChar));
                        }

                        // the case when the computer starts
                        else
                        {
                                System.out.println("Computer will start.");

                                /* how a default line looks, and the four lines of a board which are
initially set as equal to the default line,
                                WITH THE EXCEPTION that the computer makes its first move into the cell
with the coordinate (2,2)*/
                                String defaultLine = "| E | E | E | E |";        String line1 =
defaultLine;    String line2 = "| E | " + computerChoice +" | E | E |";  String line3 = defaultLine;
        String line4 = defaultLine;
                                String board = line1 + line2 + line3 + line4; // the whole board in one
line

                                // print out the initial board
                                System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);

                                /* we'll need to keep track of the following extra 2 values in the case
that the computer starts, since we want the computer
                                 * to be able to play competitively. the computer has a 100% chance of
winning if it plays the right moves corresponding
                                 * to the player's moves, and that is only possible thanks to these two
variables. */

                                int round = 0; // the current round being played
                                boolean playerPlayed2 = false; // and a boolean for if the player played
a move with the x coordinate 2.

                                while(!hasEnded(board))
                                {
                                        System.out.println("Enter coordinates:");

                                        int yCoordinate = 0;    int xCoordinate = 0;
                                        do
                                        {
                                                try
                                                {
                                                        yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();

                                                }
                                                catch (InputMismatchException e)
                                                {
                                                                System.out.println("Wrong input! Try
again:");

                                                                console.nextLine();
                                                }
                                        } while (yCoordinate == 0 || xCoordinate == 0);

                                        while ((xCoordinate > 4 || yCoordinate > 4) || board.charAt(-2 +
(4 * xCoordinate) + (17 * (yCoordinate - 1))) != 'E')
                                        {
                                                System.out.println("Wrong input! Try again:");
                                                yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
                                        }
```

```
                                        switch (xCoordinate)
                                        {
                                                case 1:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,2) + playerChoice +
line1.substring(3);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,2) + playerChoice +
line2.substring(3);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,2) + playerChoice +
line3.substring(3);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,2) + playerChoice +
line4.substring(3);
                                                        break;
                                                }
                                                break;

                                                case 2:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,6) + playerChoice +
line1.substring(7);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,6) + playerChoice +
line2.substring(7);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,6) + playerChoice +
line3.substring(7);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,6) + playerChoice +
line4.substring(7);
                                                        break;
                                                }
                                                break;

                                                case 3:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,10) + playerChoice +
line1.substring(11);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,10) + playerChoice +
line2.substring(11);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,10) + playerChoice +
line3.substring(11);
```

```
                                                break;

                                                case 4:
                                                line4 = line4.substring(0,10) + playerChoice +
line4.substring(11);
                                                break;
                                        }
                                        break;

                                        case 4:
                                        switch (yCoordinate)
                                        {
                                                case 1:
                                                line1 = line1.substring(0,14) + playerChoice +
line1.substring(15);
                                                break;

                                                case 2:
                                                line2 = line2.substring(0,14) + playerChoice +
line2.substring(15);
                                                break;

                                                case 3:
                                                line3 = line3.substring(0,14) + playerChoice +
line3.substring(15);
                                                break;

                                                case 4:
                                                line4 = line4.substring(0,14) + playerChoice +
line4.substring(15);
                                                break;
                                        }
                                        break;
                                }

                                // this is where the computer takes over to make its next move.
                                round++;

                                /* if the player did not make a move with x coordinate 2 AND it's
the first round, plays the move (3,2) and
                                 * sets playerPlayed2 to false (which will be important in
deciding for the next move) */
                                if ((xCoordinate == 1 || xCoordinate == 3 || xCoordinate == 4) &&
round == 1)
                                {
                                        line3 = line3.substring(0,6) + computerChoice +
line3.substring(7);
                                        playerPlayed2 = false;
                                }

                                /* if the player DID make a move with x coordinate 2 AND it's the
first round, plays the move (2,3) and
                                 * sets playerPlayed 2 to true */
                                else if (xCoordinate == 2 && round == 1)
                                {
                                        line2 = line2.substring(0,6) + computerChoice + " | " +
computerChoice + line2.substring(11);
                                        playerPlayed2 = true;
                                }

                                // these are the decision making processes for the final round
(which will end with the computer winning)

                                /* if the player played a move with y coordinate 1, the round is
2 and the player did not play a move with
                                 * x coordinate 2 in the initial round, plays the move (4,2) and
wins */
                                if (yCoordinate == 1 && round == 2 && playerPlayed2 == false)
                                {
```

```
                                        line4 = line4.substring(0,6) + computerChoice +
line4.substring(7);
                                }

                                /* if the player played a move with y coordinate 4, the round is
2 and the player did not play a move with
                                 * x coordinate 2 in the initial round, plays the move (1,2) and
wins */
                                else if (yCoordinate == 4 && round == 2 && playerPlayed2 ==
false)
                                {
                                        line1 = line1.substring(0,6) + computerChoice +
line1.substring(7);
                                }

                                /* if the player played a move with x coordinate 1, the round is
2 and the player DID play a move with
                                 * x coordinate 2 in the initial round, plays the move (2,4) and
wins */
                                if (xCoordinate == 1 && round == 2 && playerPlayed2 == true)
                                {
                                        line2 = line2.substring(0,14) + computerChoice +
line2.substring(15);
                                }

                                /* if the player played a move with x coordinate 4, the round is
2 and the player DID play a move with
                                 * x coordinate 2 in the initial round, plays the move (2,1) and
wins */
                                else if (xCoordinate == 4 && round == 2 && playerPlayed2 == true)
                                {
                                        line2 = line2.substring(0,2) + computerChoice +
line2.substring(3);
                                }

                                board = line1 + line2 + line3 + line4;   // updates the one-line
board string after a round is finished

                                // prints out the board after a round is played
                                System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);

                                // if at any time, there are no empty cells left on the board,
return -1 to declare a tie
                                if (!board.contains("E"))
                                {
                                        return -1;
                                }
                        }

                        // if hasEnded returns true and the while loop exits, this method returns
who has won the game (0 for player, 1 for computer)
                        return (whoWon(board, playerChoiceChar));

                }
        }

        // player decided to load a game
        else if (choice.equals("L"))
        {
                File saveFile = new File(filename); //create a File object with the filename the
user entered

                // warn the user as long as the file does not exist on disk, reprompt to enter
another filename
                while (!saveFile.exists())
                {
                        System.out.println("Invalid file. Please enter a valid file name: ");
                        filename = console.next();
```

```
                        saveFile = new File(filename);
                }

                Scanner readFile = new Scanner(saveFile); // file scanner to read from the file

                String computerChoice =  playerChoice.equals("X") ? "O" : "X";
                String playerChoiceInit = playerChoice;
                String computerChoiceInit = computerChoice;
                char playerChoiceChar = playerChoice.charAt(0);

                System.out.println("You are player " + playerChoice + " and the computer is
player " + computerChoice + "." );

                Random random = new Random();

                /* since we're now loading files, we need to run some extra tests to see who
starts instead of starting
                 * randomly. the main method checks whether there are more 'X's or 'O's in the
specified file and the starting
                 * symbol is chosen accordingly. if there's an equal number of both (including
the case where the loaded file is
                 * an empty board), the choice is made randomly */

                // assign a random number first. if the number of 'X's and 'O's are not equal,
this will get changed later anyway.
                int whoGetsToGo = random.nextInt(2);

                // if player's choice is equal to whichever one of 'X' and 'O' occurs less in the
file, they get to go first
                if (playerChoice.equals(starter))
                {
                        whoGetsToGo = 0;
                }

                // if computer's choice is equal to whichever one of 'X' and 'O' occurs less in
the file, it gets to go first
                else if (computerChoice.equals(starter))
                {
                        whoGetsToGo = 1;
                }

                // the rest of the game runs exactly the same as when the user chooses to create
a new game
                if (whoGetsToGo == 0)
                {
                        System.out.println("You will start.");

                        String line1 = readFile.nextLine();      String line2 =
readFile.nextLine();     String line3 = readFile.nextLine();      String line4 = readFile.nextLine();

                        String board = line1 + line2 + line3 + line4;

                        System.out.println(line1);       System.out.println(line2);
        System.out.println(line3);       System.out.println(line4);

                        // if at the beginning, there are no empty cells left on the board,
return -1 to declare a tie
                        if (!board.contains("E"))
                        {
                                return -1;
                        }

                        while(!hasEnded(board))
                        {
                                System.out.println("Enter coordinates:");

                                int yCoordinate = 0;     int xCoordinate = 0;
                                do
                                {
```

```
                                        try
                                        {
                                                yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
                                        }
                                        catch (InputMismatchException e)
                                        {
                                                        System.out.println("Wrong input! Try
again:");
                                                        console.nextLine();
                                        }
                                } while (yCoordinate == 0 || xCoordinate == 0);

                                while ((xCoordinate > 4 || yCoordinate > 4) || board.charAt(-2 +
(4 * xCoordinate) + (17 * (yCoordinate - 1))) != 'E')
                                {
                                        System.out.println("Wrong input! Try again:");
                                        yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
                                }

                                for (int i = 0; i < 2; i++, playerChoice = computerChoiceInit)
                                {
                                        switch (xCoordinate)
                                        {
                                                case 1:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,2) +
playerChoice + line1.substring(3);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,2) +
playerChoice + line2.substring(3);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,2) +
playerChoice + line3.substring(3);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,2) +
playerChoice + line4.substring(3);
                                                        break;
                                                }
                                                break;

                                                case 2:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,6) +
playerChoice + line1.substring(7);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,6) +
playerChoice + line2.substring(7);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,6) +
playerChoice + line3.substring(7);
                                                        break;

                                                        case 4:
```

```
playerChoice + line4.substring(7);
                                                        line4 = line4.substring(0,6) +
                                                        break;
                                                }
                                                break;

                                                case 3:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,10) +
playerChoice + line1.substring(11);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,10) +
playerChoice + line2.substring(11);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,10) +
playerChoice + line3.substring(11);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,10) +
playerChoice + line4.substring(11);
                                                        break;
                                                }
                                                break;

                                                case 4:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,14) +
playerChoice + line1.substring(15);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,14) +
playerChoice + line2.substring(15);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,14) +
playerChoice + line3.substring(15);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,14) +
playerChoice + line4.substring(15);
                                                        break;
                                                }
                                                break;
                                        }

                                        do
                                        {
                                                xCoordinate = random.nextInt(4) + 1; yCoordinate
= random.nextInt(4) + 1;
                                        }
                                        while (board.charAt(-2 + (4 * xCoordinate) + (17 *
(yCoordinate - 1))) != 'E');
                                }

                                board = line1 + line2 + line3 + line4;
```

```
                                        System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);

                                        // if at any time, there are no empty cells left on the board,
return -1 to declare a tie
                                        if (!board.contains("E"))
                                        {
                                                return -1;
                                        }

                                        playerChoice = playerChoiceInit;
                                }
                                return (whoWon(board, playerChoiceChar));
                        }

                        /* it's hard to determine a winning strategy for the computer when a file is
loaded as opposed to when
                         *  the game starts out with  an empty board. so, when a file is loaded and the
computer goes first,
                         *  it just plays randomly.*/
                        else
                        {
                                System.out.println("Computer will start.");


                                String line1 = readFile.nextLine();     String line2 =
readFile.nextLine();    String line3 = readFile.nextLine();     String line4 = readFile.nextLine();
                                String board = line1 + line2 + line3 + line4;

                                // if at any time, there are no empty cells left on the board, return -1
to declare a tie
                                if (!board.contains("E"))
                                {
                                        System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);
                                        return -1;
                                }

                                int xCoordinate = 0;    int yCoordinate = 0;
                                do
                                {
                                        xCoordinate = random.nextInt(4) + 1; yCoordinate =
random.nextInt(4) + 1;
                                }
                                while (board.charAt(-2 + (4 * xCoordinate) + (17 * (yCoordinate - 1))) !=
'E');

                                switch (xCoordinate)
                                {
                                        case 1:
                                        switch (yCoordinate)
                                        {
                                                case 1:
                                                line1 = line1.substring(0,2) + computerChoice +
line1.substring(3);
                                                break;

                                                case 2:
                                                line2 = line2.substring(0,2) + computerChoice +
line2.substring(3);
                                                break;

                                                case 3:
                                                line3 = line3.substring(0,2) + computerChoice +
line3.substring(3);
                                                break;

                                                case 4:
```

```
                                        line4 = line4.substring(0,2) + computerChoice +
line4.substring(3);
                                        break;
                                }
                                break;

                                case 2:
                                switch (yCoordinate)
                                {
                                        case 1:
                                        line1 = line1.substring(0,6) + computerChoice +
line1.substring(7);
                                        break;

                                        case 2:
                                        line2 = line2.substring(0,6) + computerChoice +
line2.substring(7);
                                        break;

                                        case 3:
                                        line3 = line3.substring(0,6) + computerChoice +
line3.substring(7);
                                        break;

                                        case 4:
                                        line4 = line4.substring(0,6) + computerChoice +
line4.substring(7);
                                        break;
                                }
                                break;

                                case 3:
                                switch (yCoordinate)
                                {
                                        case 1:
                                        line1 = line1.substring(0,10) + computerChoice +
line1.substring(11);
                                        break;

                                        case 2:
                                        line2 = line2.substring(0,10) + computerChoice +
line2.substring(11);
                                        break;

                                        case 3:
                                        line3 = line3.substring(0,10) + computerChoice +
line3.substring(11);
                                        break;

                                        case 4:
                                        line4 = line4.substring(0,10) + computerChoice +
line4.substring(11);
                                        break;
                                }
                                break;

                                case 4:
                                switch (yCoordinate)
                                {
                                        case 1:
                                        line1 = line1.substring(0,14) + computerChoice +
line1.substring(15);
                                        break;

                                        case 2:
                                        line2 = line2.substring(0,14) + computerChoice +
line2.substring(15);
                                        break;
```

```
                                        case 3:
                                        line3 = line3.substring(0,14) + computerChoice +
line3.substring(15);
                                        break;

                                        case 4:
                                        line4 = line4.substring(0,14) + computerChoice +
line4.substring(15);
                                        break;
                                }
                                break;
                        }

                        board = line1 + line2 + line3 + line4;
                        System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);

                        // if at any time, there are no empty cells left on the board, return -1
to declare a tie
                        if (!board.contains("E"))
                        {
                                return -1;
                        }

                        while(!hasEnded(board))
                        {
                                System.out.println("Enter coordinates:");

                                do
                                {
                                        try
                                        {
                                                yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
                                        }
                                        catch (InputMismatchException e)
                                        {
                                                        System.out.println("Wrong input! Try
again:");
                                                        console.nextLine();
                                        }
                                } while (yCoordinate == 0 || xCoordinate == 0);

                                while ((xCoordinate > 4 || yCoordinate > 4) || board.charAt(-2 +
(4 * xCoordinate) + (17 * (yCoordinate - 1))) != 'E')
                                {
                                        System.out.println("Wrong input! Try again:");
                                        yCoordinate = console.nextInt(); xCoordinate =
console.nextInt();
                                }

                                for (int i = 0; i < 2; i++, playerChoice = computerChoiceInit)
                                {
                                        switch (xCoordinate)
                                        {
                                                case 1:
                                                switch (yCoordinate)
                                                {
                                                        case 1:
                                                        line1 = line1.substring(0,2) +
playerChoice + line1.substring(3);
                                                        break;

                                                        case 2:
                                                        line2 = line2.substring(0,2) +
playerChoice + line2.substring(3);
                                                        break;

                                                        case 3:
```

```
                        line3 = line3.substring(0,2) +
playerChoice + line3.substring(3);
                        break;

                        case 4:
                        line4 = line4.substring(0,2) +
playerChoice + line4.substring(3);
                        break;
                }
                break;

                case 2:
                switch (yCoordinate)
                {
                        case 1:
                        line1 = line1.substring(0,6) +
playerChoice + line1.substring(7);
                        break;

                        case 2:
                        line2 = line2.substring(0,6) +
playerChoice + line2.substring(7);
                        break;

                        case 3:
                        line3 = line3.substring(0,6) +
playerChoice + line3.substring(7);
                        break;

                        case 4:
                        line4 = line4.substring(0,6) +
playerChoice + line4.substring(7);
                        break;
                }
                break;

                case 3:
                switch (yCoordinate)
                {
                        case 1:
                        line1 = line1.substring(0,10) +
playerChoice + line1.substring(11);
                        break;

                        case 2:
                        line2 = line2.substring(0,10) +
playerChoice + line2.substring(11);
                        break;

                        case 3:
                        line3 = line3.substring(0,10) +
playerChoice + line3.substring(11);
                        break;

                        case 4:
                        line4 = line4.substring(0,10) +
playerChoice + line4.substring(11);
                        break;
                }
                break;

                case 4:
                switch (yCoordinate)
                {
                        case 1:
                        line1 = line1.substring(0,14) +
playerChoice + line1.substring(15);
                        break;
```

```
                                                        case 2:
                                                        line2 = line2.substring(0,14) +
playerChoice + line2.substring(15);
                                                        break;

                                                        case 3:
                                                        line3 = line3.substring(0,14) +
playerChoice + line3.substring(15);
                                                        break;

                                                        case 4:
                                                        line4 = line4.substring(0,14) +
playerChoice + line4.substring(15);
                                                        break;
                                                }
                                                break;
                                        }

                                        do
                                        {
                                                xCoordinate = random.nextInt(4) + 1; yCoordinate
= random.nextInt(4) + 1;
                                        }
                                        while (board.charAt(-2 + (4 * xCoordinate) + (17 *
(yCoordinate - 1))) != 'E');
                                }

                                board = line1 + line2 + line3 + line4;

                                System.out.println(line1);      System.out.println(line2);
        System.out.println(line3);      System.out.println(line4);

                                // if at any time, there are no empty cells left on the board,
return -1 to declare a tie
                                if (!board.contains("E"))
                                {
                                        return -1;
                                }

                                playerChoice = playerChoiceInit;
                        }

                        return (whoWon(board, playerChoiceChar));

                }
        }

        return 0;
        }

        public static void main(String[] args) throws FileNotFoundException
        {
                Scanner console = new Scanner(System.in); // new scanner that will read from the console
                int playerWins = 0; // keeps a count of how many times the player has won
                int computerWins = 0; // keeps a count of how many times the computer has won
                String again = ""; // does the player want to play again? keeps "Y" or "N" as a string
                String starter = ""; // keeps whoever gets to go first (after counting the number of 'X'
and 'O' in the file) as a string in case the user loads a file

                System.out.println("Welcome to the XOX Game.");
                System.out.println("Would you like to load the board from file or create a new one? (L or
C)");
                String choice = console.next();

                // reads player's choice (L or C) and standardizes it to upper case
                choice = choice.toUpperCase();

                // warns user as long as they enter something other than upper case or lower case L or C
                while (!choice.equals("L") && !choice.equals("C"))
```

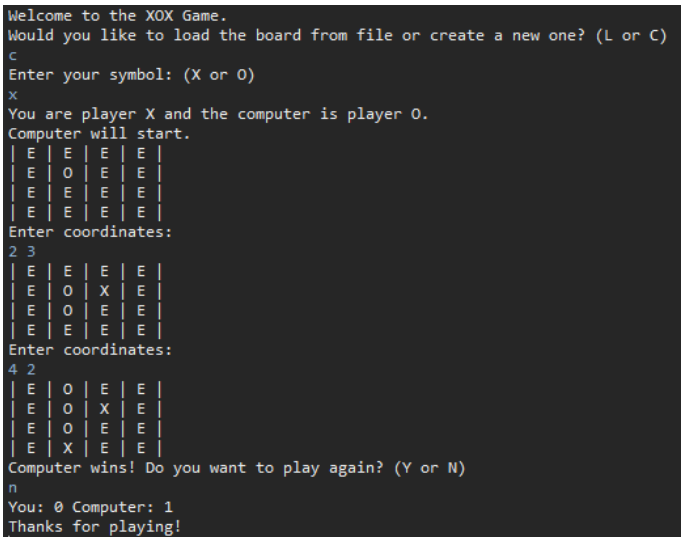
```
		{
			System.out.println("Please enter either L or C.");
			choice = console.next();
			choice = choice.toUpperCase();
		}

		String playerChoice = ""; // initializes the player's choice as an empty string (it will be assigned later after asking the player)
		String filename = ""; // initializes the filename as an empty string (it will be assigned later after asking the player)

		// player chooses to create a new file
		if (choice.equals("C"))
		{
			System.out.println("Enter your symbol: (X or O)");

			// read player's symbol choice and prompt to enter again as long as they don't enter X or O
			playerChoice = console.next();
			playerChoice = playerChoice.toUpperCase();

			while (!playerChoice.equals("X") && !playerChoice.equals("O"))
			{
				System.out.println("Please enter either X or O.");
				playerChoice = console.next();
				playerChoice = playerChoice.toUpperCase();
			}
		}

		// player chooses to load file
		else
		{
			System.out.println("Please enter the file name: ");

			// read player's filename and prompt to enter again as long as file does not exist on disk
			filename = console.next();

			File saveFile = new File(filename);

			while (!saveFile.exists())
			{
				System.out.println("Invalid file. Please enter a valid file name: ");
				filename = console.next();
				saveFile = new File(filename);
			}
			/* the following lines perform a series of tests on the loaded file to see if its format is suitable
			 * for the game. the first test (carried out by the scanner checkFirst) checks if the file contains any
			 * characters other than '|', ' ', 'E', 'O', 'X'. the second test checks whether all of the lines in the
			 * file contain exactly 17 characters (as it should for the board format specified in the assignment
			 * description.) the third test checks whether the file contains 4 lines. finally, the scanner finalCheck
			 * goes over each line in the file, counting the number of 'X's and 'O's, in order to determine who will
			 * get to go first. */
			Scanner checkFirst = new Scanner(saveFile);
			Scanner checkSecond = new Scanner(saveFile);
			Scanner checkThird = new Scanner(saveFile);
			Scanner finalCheck = new Scanner(saveFile);

			int rows = 0; // keeps the number of lines in the file
			int o = 0; // keeps the number of 'O's in the file
			int x = 0; // keeps the number of 'X's in the file
```

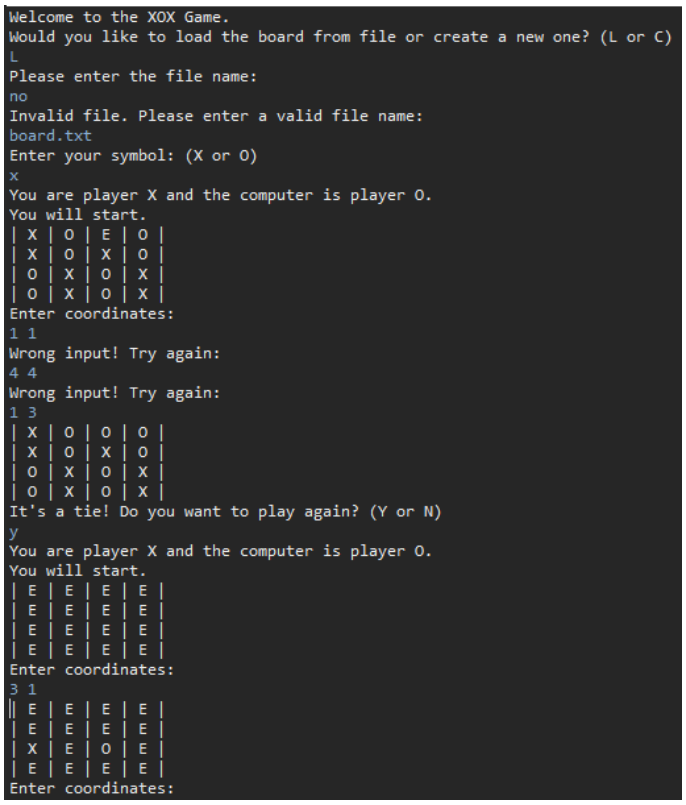
```
                        while (checkFirst.hasNextLine())
                        {
                                String checkLine = checkFirst.nextLine();
                                int lineLength = checkLine.length(); // length of the line currently
being evaluated
                                for (int i = 0; i < lineLength; i++) // counter for going over every
character in a line
                                {
                                        if (checkLine.charAt(i) != '|' && checkLine.charAt(i) != ' ' &&
checkLine.charAt(i) != 'E' && checkLine.charAt(i) != 'X' && checkLine.charAt(i) != 'O')
                                        {
                                                do
                                                {
                                                        System.out.println("Invalid file. Please enter a
valid file name: ");
                                                        filename = console.next();
                                                        saveFile = new File(filename);
                                                } while (!saveFile.exists());
                                        }
                                }
                        }


                        while (checkSecond.hasNextLine())
                        {
                                String checkLine = checkSecond.nextLine();
                                int lineLength = checkLine.length();
                                if (lineLength != 17)
                                {
                                        do
                                        {
                                                System.out.println("Invalid file. Please enter a valid
file name: ");
                                                filename = console.next();
                                                saveFile = new File(filename);
                                        } while (!saveFile.exists());
                                }
                        }


                        while (checkThird.hasNextLine())
                        {
                                checkThird.nextLine();
                                rows++;
                        }

                        if (rows != 4)
                        {
                                do
                                {
                                        System.out.println("Invalid file. Please enter a valid file name:
");
                                        filename = console.next();
                                        saveFile = new File(filename);
                                } while (!saveFile.exists());
                        }

                        while (finalCheck.hasNextLine())
                        {
                                String checkLine = finalCheck.nextLine();
                                int lineLength = checkLine.length();
                                for (int i = 0; i < lineLength; i++) // counter for going over every
character in a line
                                {
                                        if (checkLine.charAt(i) == 'O')
                                        {
                                                o++;
                                        }
                                        else if (checkLine.charAt(i) == 'X')
```

```
                                {
                                        x++;
                                }
                        }
                }

                // determines the starting symbol according to the number of 'X's and 'O's.
updates the String starter accordingly.
                if (o > x)
                {
                        starter = "X";
                }
                else if (x > o)
                {
                        starter = "O";
                }

                System.out.println("Enter your symbol: (X or O)");

                playerChoice = console.next();
                playerChoice = playerChoice.toUpperCase();

                while (!playerChoice.equals("X") && !playerChoice.equals("O"))
                {
                        System.out.println("Please enter either X or O.");
                        playerChoice = console.next();
                        playerChoice = playerChoice.toUpperCase();
                }

                Scanner readFile = new Scanner(saveFile); // Scanner that will read from the file
        }


        do
        {
                int result = playTheGame(choice, console, playerChoice, filename, starter); //
gets the return value from playTheGame (0 if player wins, 1 if computer wins, -1 if tie)
                switch (result)
                {
                        case 1:
                        System.out.println("Computer wins! Do you want to play again? (Y or N)");
                        computerWins++;
                        choice = "C";
                        again = "";
                        break;

                        case 0:
                        System.out.println("You win! Do you want to play again? (Y or N)");
                        playerWins++;
                        choice = "C";
                        again = "";
                        break;

                        case -1:
                        System.out.println("It's a tie! Do you want to play again? (Y or N)");
                        choice = "C";
                        again = "";
                        break;
                }

                // reads if the player wants to play again, saves it in the String again
                again = console.next().toUpperCase();

                // reprompts the user as long as they don't enter N or Y
                while (!again.equals("Y") && !again.equals("N"))
                {
                        System.out.println("Please enter either Y or N.");
                        again = console.next();
                        again = again.toUpperCase();
```

```
                }
            } while (again.equals("Y"));

            // displays the number of times and says goodbye
            System.out.println("You: " + playerWins + " Computer: " + computerWins);
            System.out.println("Thanks for playing!");
        }
}
```

Output of the program:





Conclusion: I believe I've correctly solved the problem and handled as many of the exceptional cases as I could think of. To be honest, I think I have some redundant parts in my code (e.g. the switch-case blocks which repeat four times) but I could not think of a better way to implement those parts. I tried to factor the switch-case blocks into a separate method but it turned out to be too challenging, so I decided to leave them as they are (in fear of completely breaking a perfectly working code just because it looks unaesthetic). I realize that the hasEnded and whoWon methods also suffer from the same unaestheticity with their huge walls of code. In my defense, in a real-world situation I would never have implemented this program using string methods only, when there are much easier means, such as arrays. I also realize that my code is extremely long, but I believe (and hope) that won't be a problem since the length of the code was not mentioned as a criterion in the assignment description.